cOOKie, a Tool for Developing RF Communication Systems for the Internet of Things

Kevin S. Miller
Montclair State University
ksmiller99@gmail.com

Christopher S. Leberknight
Montclair State University
leberknightc@montclair.edu

Abstract

There is a need for high-efficiency short-range wireless communications to connect IoT devices that have low to medium security requirements. A hardware/software tool was developed to help IoT product developers quickly and easily develop radio frequency (RF) communication systems for IoT devices where previously this was a manual, one-off process. The tool uses Software Defined Radio (SDR) and focuses on On-Off-Keying (OOK) modulation. It can be used by persons with limited knowledge of RF to analyze existing devices and capture its characteristics, which can be used to create and transmit new messages, in effect spoofing it. New device definitions can be implemented in low-cost off-the-shelf hardware for production. OOK has been found to be very efficient at binary RF communications because the transmitter is only powered when a "1" is being transmitted. This efficiency translates into a battery life of up to one year. Implementations of this system could include arrays of sensors that periodically transmit data to a traditionally-powered Internet-connected receiver. Another possible use of this system could be low-cost small transmitters to track animal movements in a defined area. Receivers placed around the area could record the time and signal strength of the transmissions. Software would be used to analyze the data and plot the animal's movements. Because the RF transmissions have a specific range, the opportunity to intercept, modify or spoof communications is highly variable. For sensitive data, rolling codes and/or public/private key encryption could be used for encoding before modulating with OOK.

1. Introduction

The motivation for the cOOKie application is based on challenges with existing applications and tools used for interfacing between SDR's and associated applications. cOOKie is an acronym for **c**# **O**n-**O**ff **K**eying **i**nteractive **e**nvironment. Ease of use is often not sufficient for many programmers leading to steep learning curves for experimenting with some of the basic applications. For example, developing and controlling an SDR to spoof ADSB signals with current applications often requires knowledge outside the scope of knowledge for many programmers. A great deal of electrical engineering domain-specific knowledge is required to create an RF communication system, but when the requirements are limited to a specific type of system, the choices are limited and can be calculated from a few inputs. Still, that required knowledge is beyond the experience of most software developers. cOOKie is a tool to help bridge that gap.

On-Off-Keying is a type of modulation that has been in use for many years for its simplicity and robustness but has recently been more closely examined for use with Internet of Things (IoT) devices.

IoT promises to have an extremely large number of devices connected to the Internet so that data can be used to make better decisions about healthcare, equipment and the environment. It is not practical to have every sensor, some of them microscopic, to be connected directly to the Internet or even to a Wi-Fi LAN. The bandwidth would not support it.

Bluetooth (BT) or Bluetooth Low Energy (BLE) are protocols for Personal Area Networks (PAN) that use the unlicensed (but not unregulated) Industrial, Scientific and Medical (ISM) band of radio frequencies. It applicable in some IoT applications where the sensor and gateway are relatively close together, however, recent advances in Bluetooth technology are closing this gap.

Zigbee is another PAN protocol in the same ISM band as Bluetooth and Wi-Fi. Zigbee devices have a range of 10 – 20 meters indoors, and up to 1500 meters outdoors with line-of-sight between devices and maximum power ratings used.

There are several trade-offs between choosing OOK, Zigbee, BT, or BLE. Of these four, OOK is the least regulated, and the most efficient, and the least expensive. It is, however, the least standardized. Unlike the other protocols, OOK is simply a modulating

URI: https://hdl.handle.net/10125/60158
ISBN: 978-0-9981331-2-6

technique – Level 1 (Physical) on the OSI model. Such a lack of standards places a larger burden on the system developer to adapt OOK to get the maximum benefits and lowest cost.

Software Defined Radio (SDR) is a hardware device that allows the user to emulate different types of Radio Frequency (RF) hardware by configuring high-speed Digital Signal Processing (DSP) hardware with software commands. Beyond emulating existing systems, new and unique systems can be defined and tested using SDR's. SDR is now becoming a feature in production communication systems that allow RF systems to be redefined "on-the-fly". Due to the expense and impracticality of upgrading systems on satellites, NASA has started to use SDR to allow RF systems to be upgraded to be compatible with systems that did not exist when the satellite was launched [12].

The cOOKie tool was written to address such issues. It was developed in C#, MATLAB, and requires a BladeRF (SDR) to analyze existing OOK systems and store the associated characteristics. These characteristics can be used to generate and transmit signals based on existing devices, or on new and unique systems designed by the user.

cOOKie is intended to be used by developers to design new OOK systems and test and modify them before committing to dedicated hardware that would much less expensive and more efficient than deploying an SDR with each installation.

The user is guided through the acquisition of signals, analyzing the signals characteristics, and saving the results for modifying the characteristics, developing new applications, and creating and transmitting new signals

2. Related Work

There is limited prior work that most closely resembles features supported by the cOOKie application. David A. Clendenen [1] used a Universal Software Radio Peripheral (USRP), which is an early SDR that is well supported, has many advanced features, but may be cost prohibitive for many experimental projects. The USRP has been the first choice of professional engineers for several years. In the development of this research, it was hoped that the techniques used in that project with the USRP could also be used with the BladeRF to develop cOOKie. However, the BladeRF is consumer/hobbyist level device [11] that is produced by a small start-up and is not as well supported.

The perceived benefit of this application is based on numerous research papers that found OOK to be the most efficient method of RF communication. When coupled with Minimum Energy (ME) coding schemes, the efficiency was increased by an order of magnitude.

OOK was compared to Bluetooth and Zigbee (802.15.4) in [2] and found to have a battery life comparable to Zigbee and better than Bluetooth. In addition OOK has been determined to have a lower cost and higher bandwidth compared to Zigbee and Bluetooth. However, there are no industry standards for OOK, which makes development without tools such as cOOKie, non-trivial.

In [3], OOK with ME coding was compared to Binary Phase-Shift Keying (BPSK) and found the former to be significantly more energy efficient when a large code-word was used but at the expense of more bandwidth being required.

Very advanced studies about using OOK with nano-networks of nano-machines inside of office spaces or even inside the human body were investigated in [7]. This highly theoretical research concluded that something like OOK with ME coding would be a good choice. While that research deals with frequencies in the Terahertz range, it does point to a possible future use of OOK and makes tools like cOOKie useful for learning to work with OOK.

3. Description of Work

Many experiments to reverse-engineer custom controllers were conducted prior to the decision to use C# and MATLAB. Work began by duplicating the ceiling fan controller published by Clayton Smith [8]. In this experiment, the RTL-SDR and GNU Radio were used to receive, demodulate, and display the signal. In place of a ceiling fan, a remote control doorbell was used. With the signal displayed on the screen, it was possible to visually count the bits and see widths of the one and zero bits. With this empirical data, it was simple to recreate it and transmit it with the bladeRF x40 without understanding anything about the data encoded in the signal. For all practical purposes, this was a simple record and playback attack. An attempt was made to create a custom flow-block for GNU radio that would extract the components of the signal, but that proved problematic due to version compatibility issues with the bladeRF driver for GNU Radio and Python. Another attempt was made with GNU Radio for Windows (GRW), but GRW required a version of the bladeRF driver that was incompatible with the Windows version used.

It was then discovered that Nuand (the bladeRF vendor) published a Simulink block for the BladeRF. MATLAB/Simulink was upgraded to the 2015b version required for the BladeRF block. However the instructions on the Nuand Wiki for using the BladeRF Simulink block produced syntax errors. Queries on the

Nuand website were not answered. The MATLAB script commands also did not work and produced errors. The Student version of MATLAB does support COM, which allows communication between Windows programs, so this was determined to be the only practical way to communicate with MATLAB from C#. There is also a way to compile MATLAB functions into a DLL that can be used in Windows applications, but this feature requires a professional version of MATLAB that was too expensive. There was a constraint that it excluded the use of professional licenses and so we were limited to non-real-time signal processing that is available with the student license.

On the Nuand forums, the lead programmer for Nuand was contacted for assistance. He suggested looking at his OOKiedokie project on GitHub. This is similar to what was planned for this project, except that it does not analyze the signals to classify them. Also, it was written in C for Linux. It was decided to port this to C# to use as a starting point for this research. C# and .NET was chosen because all .NET languages are compiled to a very fast executable that is suitable for many real-time control applications.

A C# interface to the BladeRF DLLs was found on GitHub [10]. As the porting from GCC to C# began, it quickly became obvious that because OOKieDokie relied on many standard GCC libraries that had no C# equivalent, that it would be easier to start fresh, using the MATLAB COM object for digital signal processing and C# for GUI and BladeRF API.

4. SDR Security Issues and Solutions

SDR is a powerful low-level technology tool. As such, it can be employed for eavesdropping and spoofing attacks. The difficulty and likelihood of success with these attacks is dependent on the security measures of the system being attacked and the skill and effort of the attacker. cOOKie is an off-line tool for development and analysis of RF OOK communication systems. Its vulnerability is that it is completely open to the PC that hosts it, and that PC must be secured appropriately according to the environment where it is used.

RF signals are inherently insecure and can be undetectably eavesdropped and analyzed, so the main way to secure the confidentiality of OOK data is to secure the data being transmitted by conventional methods.

To secure the integrity of the data rolling codes (described in section 7 of this paper), electronic signatures based on public/private keys, or other standard measures.

Guaranteeing availability of the data can be accomplished in several ways. Redundancy can be used if there is two-way communication. If the receiver does not respond to the signal, it can be re-transmitted until a response is received. If the communication is one-way, the signal can be transmitted at intervals. The classic example is a temperature sensor that transmits it value on a predetermined schedule.

5. Results

cOOKie was developed in C# and MATLAB. It has four tabs – Receive, Transmit, Analyze, and Device. The objective is to demonstrate how cOOKie can be used to analyze the signal from an existing device, create a model of that device, use the model to send a different signal that the receiver would recognize and process. It is also shown that the model can be edited to create a new model of a device that is more suited to the experimenter's current project's requirements.

This case study described in the following sub-sections uses a simple wireless doorbell to demonstrate the basics of how to use cOOKie to accomplish the objectives above.

5.1. Receive

The receive tab has three main areas. The Receiver Option area allows the user to configure settings on the SDR for receiving signals. There is a drop-down list at the top that allows the user to select one of the supported SDR models. At this time, only the BladeRF is supported. When the SDR model is selected, the appropriate DLLs are loaded and the current settings are retrieved from the SDR and loaded into the form. Other information about the SDR, such as hardware version, serial number, and the BladeRF, FPGA size and version is also displayed.

The receive settings for the SDR can be changed. When exiting a setting input, that setting is sent to the SDR. If the SDR does not accept a setting, one of two things can happen depending on the SDR. There may be an error message from the SDR in the SDR Status window or, as in the case with the BladeRF, the SDR will select the closest valid setting and return that value to the input field.

The top-right area of the receive tab allows the user to select details about how the received data should be saved. It is strongly suggested that the name chosen for the file includes the sample rate since that value will be required to do the analysis. The name of the file to save the data in is selected here. There are options to save the data as binary or text, unfiltered or filtered, and an

option to select which filter. At this time, only binary and unfiltered are supported. The receive tab with typical settings is shown in figure 1.

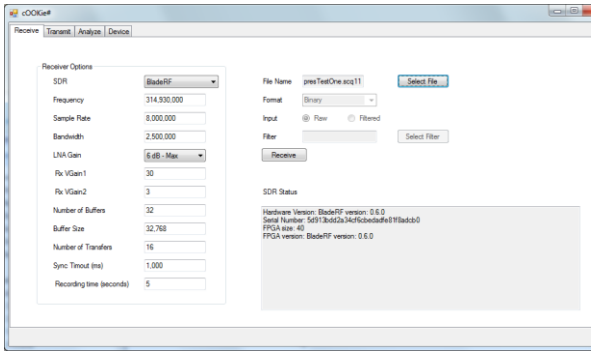

Figure 1 Receive settings

When the receive button is clicked the SDR begins to receive using the settings, and when the selected time is reached, the file is saved which is indicated in the log window.

5.2. Analyze

The Analyze tab allows the saved signals to be viewed, filtered, and analyzed. Select a file that was previously saved using the receive tab. Clicking on the "Filter in MATLAB" button before any filter data is input will show the raw signal, as in figure 2. The FIR (Finite Impulse Response) filter order is input. It is a number between 1 and 128 and represents how many stages the filter should have. More stages produce sharper edges passband edges but are slower to process. Since this is not a real-time system, it is suggested that 128 stages be used.

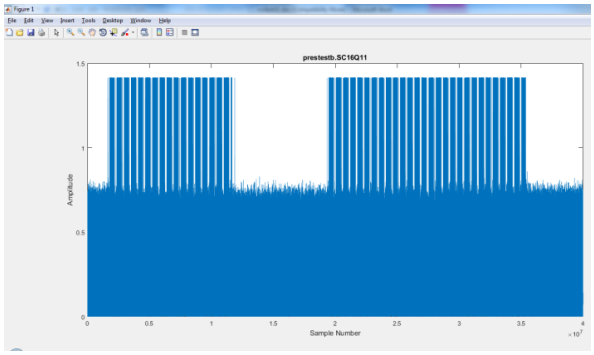

Figure 2 Raw signal after reception

The goal with filtering is to get a clean, rectangular waveform of a single word (and the first start bit of the next word) that can be analyzed. In Figure 2, there are 40 million samples, as indicated by the x scale. The signal was from a wireless doorbell, whose button was pressed twice and held for a moment each time. Each of the vertical spikes is a word. Figure 3 shows the results of filtering and isolating the first word (and next start bit) of the raw signal.

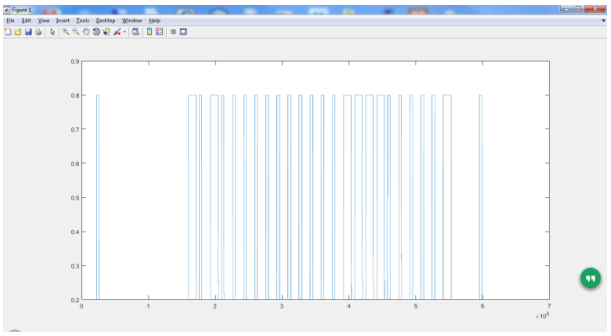

Figure 3 A single word after filter and zoom

After filtering and zooming, the word can be analyzed by pressing the Analyze button. The figure below shows the filter and zoom inputs as well as the results of the analysis. The analysis tab has a radio button that allows the user to select to view the results as sample numbers, frequency or time. Depending on how the user is using the data, one format may be more convenient.

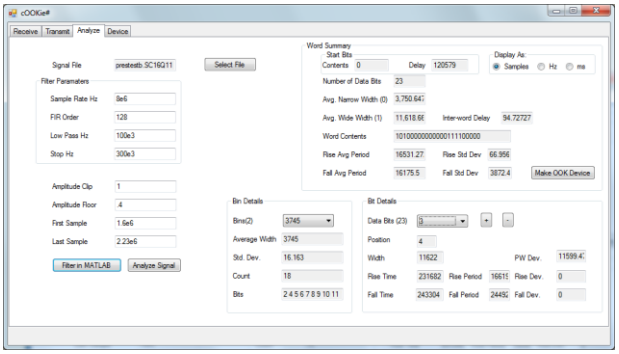

Figure 4 Filter & zoom inputs, analysis results

Figure 5 explains the various elements of the signal analysis.

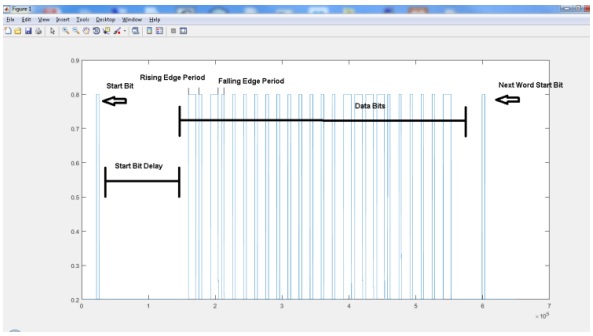

Figure 5 Analysis elements

After analysis, the characteristics can be saved as a "device" by pressing the "Make OOK Device", which opens the "Device" tab and inserts the characteristics from the analysis tab. Pressing the "Save" button will save the device characteristics as a JSON file that can be recalled at any time. It is also possible on this tab to alter any of the characteristics or to create a new device from scratch.

A new signal can be generated from the Devices tab by pressing the "Make Signal" button. The contents of the word can be modified if desired. The signal is saved as an SC16Q11, like the received signals.

On the Transmit tab, any of the generated signals can be transmitted by selecting the file. The generated signals are the intermediate RF signals and can be transmitted on any carrier frequency. If you want to trigger the same device that this signal was based on, be sure to select the same carrier frequency and sample rate. Figure 6 shows the transmit tab.

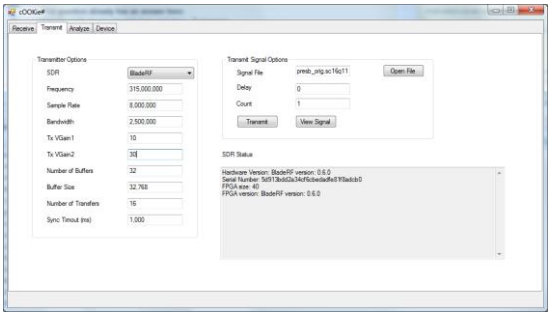

Figure 6 Transmit

6. Conclusion and Future Work

This tool makes it easier for non-engineers create radio communication systems, but there are several areas for improvement. If the user has a device but does not know what frequency is being used, another tool is needed to discover the frequency. Currently, the Fast Fourier Transform display in SDR# is being used. A new tab with an FFT display would also be helpful.

An automated filtering feature would also be helpful. After the center frequency is known, a reasonable guess can be made about the bandwidth of the signal. The program can then try a range of filter options until the signal has the sharpest edges and the least noise.

Support for other SDRs could be accomplished by abstracting the functions used by cOOKie and creating an interface to that contain the methods used.

A few changes can turn cOOKie into an RF cybersecurity tool that can jam and spoof OOK signals. If the device is pre-selected and the SDR is placed into streaming mode, signals can be filtered, and demodulated in real-time. The major obstacle for this is that the Student version of MATLAB is being used, which does not support compiling or streaming.

7. SDR as an Attack Device

There are other ways that an SDR can be used to attack RF OOK systems.

7.1 Signal Deletion

This sophisticated attack can only be accomplished in certain specific circumstances. The attacking SDR must be placed between the attacked system's transmitter and receiver. The SDR's FPGA must be programmed for this task because its response to the attacked signal must be as close to instantaneous as possible. In this scenario, the SDR's receiver is configured to recognize only certain types of signals. The instant that the signal is recognized, the transmitter can begin to transmit on the same frequency and bit rate as the attacked signal to "jam" the transmission and prevent the attacked signal from being received. The target system may be designed to detect such jamming. A more sophisticated version of this type of attack, the SDR can monitor the attacked signal and transmit an exact inverted copy of the attacked signal in near-real-time, effectively deleting it from the air. In both of these attacks, a spoofed signal containing different data could be transmitted to the attacked receiver.

7.2 Countermeasures

One way to detect these kinds of attacks would be to have multiple receivers in multiple positions. Although it is possible that none of the receivers could receive a usable signal, the attack would be obvious and by using a high resolution timer and synchronized clocks on the receivers, the attacker's position can be determined in a way that is similar to how GPS receivers can locate themselves based on the timing of signals from satellites.

Rolling codes is another countermeasure for this type of attack. In this countermeasure there is a pseudo-random number at the beginning of the signal. The receiver compares that number to the expected number and if it matches, the rest of the signal is processed. The receiver and the transmitter both use the same algorithm to hash the code signal and the output of that hash is the next preamble. Prior to using this method, the transmitter and receiver must be synchronized, usually by putting the receiver in a "learning" mode, where it takes the next signal it receives as the seed for the next signal. Because signals are sometimes not received for various reasons, the transmitter and receiver can get out of sync. The common way to deal with this is for the receiver to create a table with an arbitrary number of records that begin with next expected preamble and each succeeding record containing the hash of the previous preamble. If any of the preambles on that table are received it is assumed that it is correct, the signal is processed and a new table is created with the most recent preamble as the seed.

8. References

[1] D. Clendenen, "A software defined radio testbed for research in dynamic spectrum access", 2012.

[2] M. Prashanth Holenarsipur, "I'm OOK. You're OOK? | EE Times", *EETimes*, 2016. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1276362. [Accessed: 19- Nov- 2016].

[3] Q. Tang, S. Gupta and L. Schwiebert, "BER Performance Analysis of an On-off Keying based Minimum Energy Coding for Energy Constrained Wireless Sensor Application", *PeS*, vol. 1, p. 2.

[4] Y. Prakash and S. Gupta, "Energy efficient source coding and modulation for wireless applications", 2003, pp. 212--217.

[5] J. Kirkhorn, "Introduction to IQ-demodulation of RF-data", *IFBT, NTNU*, vol. 15, 1999.

[6] M. Zainuddin, E. Dedu and J. Bourgeois, "Low weight code comparison for electromagnetic wireless nanocommunication", 2012.

[7] I. Akyildiz and J. Jornet, "The internet of nano-things", *IEEE Wireless Communications*, vol. 17, no. 6, pp. 58--63, 2010.

[8] C. Smith, "Reverse engineering a ceiling fan – Clayton's Domain", *Irrational.net*, 2016. [Online]. Available: http://www.irrational.net/2014/03/22/reverse-engineering-a-ceiling-fan/. [Accessed: 19- Nov- 2016].

[9] J. Szymaniak, "jynik/OOKiedokie", *GitHub*, 2016. [Online]. Available: https://github.com/jynik/OOKiedokie. [Accessed: 20- Nov- 2016].

[10] J. Picod, "jmichelp/sdrsharp-bladerf", *GitHub*, 2016. [Online]. Available: https://github.com/jmichelp/sdrsharp-bladerf. [Accessed: 20- Nov- 2016].

[11] "bladeRF - USB 3.0 Software Defined Radio", *Kickstarter*, 2016. [Online]. Available: https://www.kickstarter.com/projects/1085541682/bladerf-usb-30-software-defined-radio. [Accessed: 21- Nov- 2016].

[12] R. Reinhart, S. Johnson, T. Kacpura, C. Hall, C. Smith and J. Liebetreu, "Open Architecture Standard for NASA's Software-Defined Space Telecommunications Radio Systems", *Proceedings of the IEEE*, vol. 95, no. 10, pp. 1986--1993, 2007.